AF/2124

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Group Art Unit: 2183
Examiner: William H. Wood

In re Application of:	Thomas D. Hartnett et al.
Serial No.:	09/468,051
Filed:	December 20, 1999
Docket No.:	RA-5271K
Title:	PIPELINE CONTROLLER FOR PROVIDING INDEPENDENT EXECUTION BETWEEN THE PRELIMINARY AND ADVANCED STAGES OF A SYNCHRONOUS PIPELINE
Customer No.:	027516

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P O Box 1450
Alexandria, VA 22313-1450

SUPPLEMENTAL APPEAL BRIEF TRANSMITTAL

CERTIFICATE UNDER 37 C.F.R. 1.8: I hereby certify that this correspondence is being deposited with the United States Postal Service on the date shown below with sufficient postage as First Class Mail in an envelope addressed to: Mail Stop Appeal Brief – Patents, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450 on this 10th day of February, 2005.

By M. A. Hubbard
M. A. Hubbard

Sir:

Transmitted herewith is:

- A request for reinstatement of the Appeal filed July 16, 2004 pursuant to 37 CFR 1.193(b)(2).

- Appellant's Supplemental Appeal brief filed in triplicate pursuant to 37 C.F.R. 1.192(c);

- ☑ Applicant petitions for an extension of time under 37 C.F.R. 1.136 (fees: 37 C.F.R. 1.17(a)-(d)) for the total number of months checked below:

02/15/2005 AWONDAF1 00000053 193790 09468051

01 FC:1251 120.00 DA

Extension (months)	Fee for other than small entity
☑ one month	$ 110.00
☐ two months	$ 430.00
☐ three months	$ 980.00
☐ four months	$1,530.00
☐ five months	$2,080.00

- It is believed no further fee is required, since the fee paid for the appeal brief filed on July 16, 2004 will be applied to this request for reinstatement of appeal on the same application pursuant to MPEP 1208.03.

FEE PAYMENT

Please charge Account No. 19-3790 the sum of $ 110.00. If any additional extension, and/or fee are required, please charge Account No. 19-3790.

A duplicate of this transmittal is attached.

Respectfully submitted,

Beth McMahon

Beth McMahon
Reg. No. 41,987
Phone.: (651) 635-7893
Unisys Corporation
MS 4773
P.O. Box 64942
St. Paul, MN 55164-0942

APPEAL TO THE BOARD OF PATENT APPEALS AND INTERFERENCES
Group Art Unit: 2183
Examiner: William H. Wood

Customer Assignment No.: 027516 February 10, 2005
Serial No.: 09/468,051
Filed: December 20, 1999
In re Application of: Thomas D. Hartnett, John S. Kuslak, and Gary J. Lucas
Title: PIPELINE CONTROLLER FOR PROVIDING INDEPENDENT EXECUTION BETWEEN THE PRELIMINARY AND ADVANCED STAGES OF A SYNCHRONOUS PIPELINE
Docket No.: RA-5271K

REQUEST FOR REINSTATMENT OF APPEAL AND APPELLANT'S SUPPLEMENTAL APPEAL BRIEF PURSUANT TO 37 C.R.F. §1.193(b)(2) and §41.37

MS Appeals - Patents
Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

This brief is being submitted within four (4) months of receiving a non-final Office Action dated 10/18/2004 on the merits setting a three (3) month period for reply, and re-opening prosecution of the Claims after Appeal under 37 CFR §1.193(b)(2) and MPEP §1208.02. A petition for a **ONE-MONTH EXTENSION** of time under 37 CFR 1.136(a) is attached herewith.

It is believed that this supplemental brief, which is filed in triplicate, need not be accompanied by an additional fee because the fee paid for the original appeal in the subject case is applied to this request for reinstatement of the appeal. Permission is hereby granted to charge deposit account number 19-3790 for any errors in fee calculation. Appellants request this Supplemental Appeal Brief be made of record and fully considered.

CERTIFICATE OF MAILING (37 CFR 1.8(a))
I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Box MS Appeals - Patents, Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date shown below:

Angie Hubbard **February 10, 2005**
(Angie Hubbard ***(Date)***

TABLE OF CONTENTS

Real Party In Interest

The real party in interest is Unisys Corporation, with an address as follows:

Unisys Corporation
Township Line and Union Meeting Roads
Blue Bell, Pennsylvania 19424

Unisys Corporation is the real party in interest through an assignment from all of the inventors of their entire interest, recorded on 12/20/1999 in the USPTO at Reel/Frame 010650/0869.

Related Pending Appeals or Interferences

There are no pending appeals or interferences related to the subject Appeal.

Status of Claims

Claims 1-7 and 21-46 remain pending, and stand finally rejected.

Claims 1-7 and 21-46 were presented for appeal in an Appeal Brief filed by Applicants on July 16, 2004. In the Office Action mailed October 18, 2004, the Examiner reopened prosecution of each of the Claims 1-7 and 21-46 presented for appeal, citing new grounds of rejection while maintaining the previous grounds of rejection. Reinstatement of the Appeal of each of Claims 1-7 and 21-46 is hereby requested.

Status of the Amendments

A first Amendment was submitted on October 25, 2002 to amend Claims 1, 7, 8 and 14 of originally presented Claims 1-20. This amendment has been entered.

A second amendment, which was filed April 30, 2003 and accompanied a Request for Continuing Examination (RCE), amended Claim 1, canceled Claims 8-20, and added Claims 21-46. This amendment has been entered.

No further amendments have been submitted or entered.

A clean copy of Claims 1-7 and 21-46, as amended, is provided as Appendix A. Claims 1 and 7, which were modified by entered amendments, are designated as "Previously Amended". Claims that have not been amended, including Claims 2 through 6, are labeled "Original". The remaining Claims 21-46, which were presented in the RCE filed April 30, 2003, are marked "Previously Added".

Summary of the Claimed Subject Matter

Independent Claim 1

Claim 1 provides:

1. For use in an instruction processor that executes instructions included in a predetermined instruction set at an execution rate determined by a system clock signal, a synchronous instruction pipeline, comprising:

a pipeline execution circuit to process a first predetermined number of instructions simultaneously, each of said first predetermined number of instructions being in a respectively different stage of execution within said pipeline execution circuit, instructions being capable of advancing to a next stage of execution within said pipeline execution circuit at a time determined by the system clock signal; and

a pipeline fetch circuit coupled to provide each of the first predetermined number of instructions directly to said pipeline execution circuit, the pipeline fetch circuit to retain a second predetermined number of instructions simultaneously, each of said second predetermined number of instructions being in a respectively different stage of processing within said pipeline fetch circuit, an instruction being capable of advancing to a next stage of execution within said pipeline fetch circuit at a time determined by the system clock signal and independently of the times at which instructions advance to a next stage of execution within said pipeline execution circuit.

The invention of Claim 1 provides an improved synchronous instruction pipeline system and method for a data processing system. This instruction pipeline includes fetch and execution stages that are *independently operable*, yet *directly interconnected*.

Applicants' system can best be understood through a general discussion of pipeline architectures. As is known in the art, an instruction pipeline includes multiple logic sections, or "stages", each of which performs one of the operations required to process an instruction to completion. After an instruction enters the pipeline, it is processed in an "assembly line" fashion, moving from one stage to the next until instruction execution is completed. At each stage, a task such as instruction decode, operand address generation, or the fetching of one or more operands, is initiated.

According to the current invention, an instruction pipeline is provided that includes a first number of fetch stages and a second number of execution stages. In an exemplary embodiment, the fetch stages include stages 0Y through 3Y, and the execution stages comprise stages 1X through 6X.[1] The fetch stages 0Y through 3Y, which are shown implemented by elements 111, 38, 116, and 118 of Figure 7, control the operations that must occur before signals are provided to the execution logic of the instruction processor. These operations include the fetching and decoding of the instruction.

After an instruction has been processed by each of the fetch stages, it is provided to the execution stages, which control operations such as operand address generation, operand retrieval, arithmetic processing, and the storing of any results.[2] These execution stages are implemented by elements 36, 124, 138, 150, 160, 164 of Figure 7.

When Applicants' instruction pipeline is operating in a fully overlapped manner, each of the fetch and execution stages is processing a different instruction at a given time, as is shown in Figure 2. In this case, the instructions move from one pipeline stage to the next in lock-step with one another. Because the pipeline is synchronous, this movement occurs during each cycle of the system clock.

Some situations can disrupt the type of fully overlapped processing discussed above. For instance, sometimes an instruction may "stall" within an

[1] Applicants' Specification ("Specification"), page 12 line 5 - page 13 line 3 in reference to Figure 1.
[2] Applicants' Specification ("Specification"), page 6 lines 11-17.

execution stage of the pipeline so that the instruction is no longer advancing. As an example, Applicants' pipeline is capable of executing "extended-mode" instructions that require more than one clock cycle to complete the first of the execution stages. When an extended-mode instruction resides within the first execution stage, subsequent execution stages become empty because instructions have ceased to advance into these stages.[3] This is depicted in Applicants' Figure 6. As another example, if an attempt to obtain an operand from an operand cache memory results in a cache miss, instruction execution is delayed while the operand is retrieved from a slower memory. This also causes one or more downstream execution stages to empty.[4]

In a manner similar to that discussed above with respect to the execution stages, one or more of the fetch stages may become empty. For example, a cache miss may occur during the retrieval of an instruction, as is shown in Applicants' Figure 8. This will cause a delay in instruction availability so that some of the fetch stages do not contain an instruction. This may also occur when a flush operation must be initiated to clear a sequence of instructions from the fetch stages of the pipeline after a branch misprediction occurs.[5]

To optimize operation of the data processing system during the types of situations discussed above, Applicants' pipeline <u>decouples</u> the fetch stages from the execution stages. This allows instructions to enter into, and advance between, the fetch stages of the pipeline regardless of whether instructions are

---

[3] Specification page 6 line 26 through page 7 line 7.
[4] Specification page 8 lines 15-21.
[5] Specification page 9 lines 9-16.

advancing between the execution stages.[6] This may be illustrated by the following example. Consider a situation wherein a cache miss occurs while one of the fetch stages is retrieving an instruction. While waiting for this instruction to be returned from a slower memory, instructions stored within the downstream pipeline stages continue to advance within the pipeline. This causes some of the downstream fetch stages to become empty. During this time, an extended-mode instruction enters the first execution stage. As previously discussed, this type of instruction will reside within the first execution stage for multiple clock cycles. This will prevent subsequent instructions in the instruction stream from entering the execution stages of the pipeline until the first stage of execution has completed for the extended-mode instruction.

In a situation such as that described above, the decoupling capability provided by Applicants' pipeline optimizes instruction throughput. According to this invention, instruction fetching is allowed to continue even though no additional instructions are entering the execution stages. Thus, in the current example, when the instruction that resulted in the cache miss finally becomes available from memory, that instruction may enter, and advance between, the various fetch stages. All decode operations may be completed for this instruction so that it is available to enter the first of the execution stages after processing has completed for the extended-mode instruction[7]. Moreover, additional instructions may be retrieved from memory while the extended-mode

[6] Specification page 7 lines 8-11.
[7] Specification page 8 lines 10-12.

instruction completes the first execution stage.[8] These additional instructions are allowed to fill any of the fetch stages that have emptied because of the cache miss situation.

The control circuit that decouples the fetch and execution stages in the above-described manner is shown in Applicants' Figures 10 and 11. The manner in which instructions are allowed to advance between the fetch stages of the pipeline <u>independently</u> of the execution stages is determined by the state of various control signals shown in Figures 10 and 11, as is described on pages 20 through 25 of Applicants' Specification[9].

As can be appreciated from the foregoing discussion, the decoupling of the fetch and execution stages increases the chances that an instruction will be available for entry into the first stage of execution when a previous instruction has completed this stage. It further ensures that overlapped execution will resume as quickly as possible after a stall has occurred in an execution stage or after a fetch stage has emptied. If the fetch and execution stages are not decoupled in this manner, the stalling of an execution stage will cause the fetch stages to likewise stall so that empty fetch stages remain empty. When instructions are once again allowed to advance within the execution stages, some fetch stages remain empty so that the system is not operating in a fully

---

[8] Specification page 8 lines 5-8.
[9] Specification page 20 line 22 through page 25 line 2. The control signals of Figures 10 and 11 include the valid sequence signal 230, hit signal 224 and valid instruction signal 228 of Figure 10, and the 1Y window signal 236, 2Y valid signal 244, 3Y valid signal 251, instruction dispatched signal 246, and flush signal 248 of Figure 11.

overlapped manner. Therefore, if fetch and execution stages remain decoupled, processing throughput is diminished.

The foregoing discussion focuses on the ability of Applicants' invention to decouple fetch and execution stages of a pipeline. It is important to note that this capability is provided while at the same time allowing the fetch and execution stages to remain *directly interconnected* to one another. This means that the last fetch stage "3Y" is interconnected to provide instructions directly to the first "1X" execution stage.[10] This is shown in Applicants' Figures 7 and 11. In particular, Figure 11 illustrates decode logic 260 of fetch stage 3Y directly coupled to provide hardware signals on lines 122 to instruction decode dispatch logic 124 of execution stage 1X, which is illustrated in Figure 7. Stage 1X may then dispatch the signals to other logic sections of the IP for execution[11].

By providing hardware signals directly between the last fetch stage 3Y to the first execution stage 1X in the foregoing manner, execution is allowed to begin within the first execution stage without incurring any unnecessary delay. This is shown in the timing diagram of Figure 5, wherein no delay occurs between the pre-decode operations of the last fetch stage 3Y, represented by line 86, and the decode and dispatch operations of the first execution stage 1X, represented by line 88.

One alternative approach to Applicants' system for directly interconnecting fetch and execution stages involves *indirectly* connecting these

[10] See, for example, Figure 11, which illustrates the decode logic 260 of stage 3Y coupled to provide instructions on line 122 to the instruction decode dispatch logic 124 of Figure 7, which performs the 1X execution stage.

stages via a storage device. For example, a queue may be inserted between the last fetch stage and the first execution stage. According to this method, instructions that are awaiting entry into the first execution stage are stored into the queue by the last fetch stage. When the first execution stage becomes available to receive another instruction, it retrieves the next instruction from this queue. While this alternative mechanism provides a simple approach to decoupling the fetch and execution stages, it imposes delay within the pipeline. This is particularly true in cases wherein the time to store, then retrieve, an instruction cannot be made transparent to the pipeline.

Rather than utilize a storage device in the manner discussed above to decouple pipeline stages, Applicants' system employs the control circuit described above to perform this task.[12] This control circuit allows instructions to advance between fetch stages of a pipeline even if an instruction is stalled within the execution stages, while further eliminating the need to utilize a storage device to couple the fetch and execution stages of the pipeline, as is done in some prior art systems.

The foregoing discussion may be summarized in reference to the specific language of Claim 1, which provides:

"...a synchronous instruction pipeline[13], comprising:

---

[11] Specification page 17 lines 4-10.
[12] See Applicants' Figures 10 and 11.
[13] Specification page 14 line 3 through line 27 describes an embodiment of Applicants' instruction pipeline in regards to the timing diagram of Figure 5. The circuit that implements this pipeline is shown in Figure 7.

a pipeline execution circuit[14] to process a first predetermined number of instructions...each... being in a respectively different stage of execution[15]...; and

a pipeline fetch circuit[16] coupled to _provide_ each of the first predetermined number of instructions _directly_[17] to said pipeline execution circuit, the pipeline fetch circuit to retain a second predetermined number of instructions... each ... being in a respectively different stage of processing within said pipeline fetch circuit[18], an instruction being capable of _advancing_ to a next stage of execution _within said pipeline fetch circuit...independently of_ the times at which instructions advance to a next stage of execution within _said pipeline execution circuit_. [19]"

[14] Specification page 17 line 4 through page 18 line 16 describes the execution circuit in regards to Figure 7. This circuit includes the Instruction Decode Dispatch Logic 124, Main Control Store 138, Operand First-Level Cache (O-FLC) 36, Operand Address Generate Logic 150, Addressing Environment Logic 160, and Arithmetic Logic 164.
[15] In the current implementation, six execution stages 1X through 6X are provided, each to process a respective instruction at a given time, such that six instructions may be executing simultaneously. These exemplary stages are shown in Figure 1 as the Decode Dispatch Stage 1X, Operand Address Generation Stage 2X, and so on. See Specification page 12.
[16] Specification page 16 line 9 through page 17 line 3 describe the fetch circuit in reference to Figure 7. This circuit includes the Instruction Address Generation Section 111, Instruction First-Level Cache (I-FLC) 38, Instruction Queue Logic 116, and 2Y/3Y Pipeline Logic 118.
[17] Interfaces 122 and 128 of Figures 7 and 11 directly couples the 3Y stage of the 2Y/3Y Pipeline Logic 118 to the Instruction Decode Dispatch Logic 124, which implements the first execution stage 1X. This is described on page 17 lines 4 through 8 of the Specification.
[18] In the current implementation, four fetch stages 0Y through 3Y are provided, each to process a respective instruction at any given time. These exemplary stages are shown in Figure 1 as the Address Generation Stage 0Y, Read Instruction Stage 1Y, and so on. See Specification page 12 lines 11 through 16.
[19] See, for example, Specification page 20 line 22 through page 25 line 2. The control signals of Figures 10 and 11 include the Tag Hit Signal 224, Valid Instruction Signal 228, and Valid Sequence Signal 230 of Figure 10, and the Valid Bit Indicator 244, Instruction Dispatched Signal 246, Flush Signal 248 and Feedback Path 251 of Figure 11.

To summarize, the claimed pipeline of Claim 1 provides a system wherein the fetch stages are directly coupled to, yet operate independently of, the execution stages. The remaining independent Claims are similar to Claim 1, as follows:

Independent Claim 21

Claim 21 provides:

"...a synchronous pipeline circuit, comprising:

an execution circuit to provide a first predetermined number of execution stages, each being capable of performing a respective processing operation on a respective instruction; and

a fetch circuit ...to provide a second predetermined number of fetch stages, each ...capable of performing a respective pre-execution operation on a respective instruction, the fetch circuit to *transfer each instruction* ...*directly*... to one of the execution stages, ...*instructions ... advancing* to different available *fetch stages independently of* whether instructions are advancing within *the execution stages*."

The citations to the Specification and Drawings provided in reference to Claim 1 above apply to the similar elements of Claim 21.

Independent Claim 27

Claim 27 provides:

"a synchronous instruction pipeline... comprising:

a first predetermined number of fetch stages to simultaneously process at least a first predetermined number of instructions;

a second predetermined number of execution stages to simultaneous process a second predetermined number of instructions... received *directly from* one of the fetch stages; and

... the fetch stages ...providing an instruction to a different one of the fetch stages... *irrespective of movement*...*between the execution stages*."

The citations to the Specification and Drawings provided in reference to Claim 1 above apply to the similar elements of Claim 27.

Independent Claim 32

Claim 32 recites:

"A synchronous instruction pipeline...comprising:

an execution circuit having a first predetermined number of execution stages to execute a first predetermined number of instructions substantially simultaneously; and

a fetch circuit having a second predetermined number of fetch stages to perform pre-execution operations...the fetch stages ...to _provide each instruction_..._directly to_ one of the _execution stages_,.... instructions being capable of advancing between... the fetch stages _regardless_ of whether an _instruction is being transferred by the fetch circuit to the execution circuit_."

The citations to the Specification and Drawings provided in reference to Claim 1 above apply to the similar elements of Claim 32.

It may be noted that Claim 32 describes the independent operability of the fetch and execution circuits in a different way than does Claim 1. In particular, Claim 32 recites that instructions are capable of advancing between the fetch stages regardless of whether the execution circuit is no longer receiving additional instructions from the fetch circuit. The execution circuit no longer receives instructions from the fetch circuit when the movement of instructions between the stages of the execution circuit has temporarily stalled.

Independent Claim 40

"A method of processing instructions within a synchronous pipeline..., comprising:

a.) performing pre-execution operations on a first predetermined number of instructions ...within a first predetermined number of fetch stages...;

b.) executing a second predetermined number of instructions ...within a second predetermined number of execution stages...each of the second ...instructions were *received directly* from one of the fetch stages; and

c.) allowing ...the first predetermined number of instructions to *advance between ...fetch stages independently of* ...the second predetermined number of instructions...*advancing between ones of the execution stages*."

The citations to the Specification and Drawings provided in reference to Claim 1 above apply to the similar elements of Claim 40.

Independent Means-Plus-Function Claim 46

Claim 46 provides:

"A pipeline circuit[20]...comprising:

instruction fetch means for performing pre-execution operations on a first predetermined number of instructions substantially simultaneously[21]...;

instruction execution means for executing a second predetermined number of instructions substantially simultaneously[22]...each of the second ...instructions ...*received directly*[23] from ...the fetch stages; and wherein

the instruction fetch means includes *means for allowing...instructions to advance ...irrespective of whether instructions are advancing within the execution stage*. [24]"

Before continuing, it may be noted that the foregoing discussion of the subject matter of the independent Claims pursuant to 37 C.F.R. § 41, 37(c)(1)(v) identifies representative subject matter for the various claimed elements. However, the abundance of supporting subject matter in the application prohibits

[20] Specification page 14 line 3 through line 27 describes an embodiment of Applicants' instruction pipeline in regards to the timing diagram of Figure 5. The circuit that implements this pipeline is shown in Figure 7.
[21] See, for example, Specification page 16 line 9 through page 17 line 3 in reference to Figure 7 instruction Address Generation Section 111, Instruction First-Level Cache (I-FLC)38, Instruction Queue Logic 116, and 2Y/3Y Pipeline Logic 118.
[22] See, for example, Specification page 17 line 4 through page 18 line 16 in regards to Figure 7, instruction decode dispatch logic 124, main control store 138, Operand First-Level Cache (O-FLC) 36, operand address generate logic 150, addressing environment logic 160, and arithmetic logic 164.
[23] Interfaces 122 and 128 of Figures 7 and 11 that couple 2Y/3Y Pipeline Logic 118 to 1X Instruction Decode Dispatch Logic 124, as described on page 17 lines 4-8.
[24] See, for example, Specification page 20 line 22 through page 25 line 2. The control signals of Figures 10 and 11 include the Tag Hit Signal 224, Valid Instruction Signal 228, and Valid Sequence Signal 230 of Figure 10, and the Valid Bit Indicator 244, Instruction Dispatched Signal 246, Flush Signal 248 and Feedback Path 251 of Figure 11.

identifying all textual and diagrammatic references to each claimed recitation. Appellant thus submits that other subject matter not specifically identified within the foregoing discussion is set forth in the Specification to support the Claim elements.

Issues

I. Whether Claims 1, 21, 27, 32, 39, 40 and 46 are unpatentable under 35 USC §103(a) over U.S. Patent Number 6,112, 295 to Bhamidipati et al. (hereinafter "Bhamidipati") in view of Hayes, John P., "Computer Architecture and Organization", (hereinafter "Hayes").

This issue may be divided into the following sub-issues A.) - C.):

A.) Whether the Bhamidipati elements cited by the Examiner render Claims 1, 21, 27, 32, 39, 40 and 46 unpatentable in view of Hayes.

B.) Whether any other elements of Bhamidipati render Claims 1, 21, 27, 32, 39, 40 and 46 unpatentable in view of Hayes.

C.) Whether the Examiner has set forth a prima facie case for obviousness with respect to Claims 1, 21, 27, 32, 39, 40 and 46.

II. Whether Claims 1, 3-4, 21, 27, 32, 39, 40 and 46 are anticipated under 35 USC § 102(e) by Bhamidipati.

III. Whether Claims 2-6, 22-25, 28-30, 33-38, 41-43, and 45 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes, and further in view of U.S. Patent Number 6,026,477 to Kyker et al. (hereinafter, "Kyker").

IV. Whether Claims 2, 5-6, 22-25, 28-30, 33-38, 41-43, and 45 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Kyker.

V. Whether Claim 7 is unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes and Kyker, and further in view of U.S. Patent Number 5,577,259 to Alferness et al. (hereinafter, "Alferness").

VI. Whether Claim 7 is unpatentable under 35 USC §103(a) over Bhamidipati in view of Kyker, and further in view of U.S. Patent Number 5,577,259 to Alferness et al. (hereinafter, "Alferness").

VII. Whether Claims 26, 31, and 44 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes and further in view of Alferness.

VIII. Whether Claims 26, 31, and 44 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Alferness.

Grouping of the Claims

Claims 1-6, 21-25, 27-30, 32-43, and 45-46 stand or fall together; and

Claims 7, 26, 31, and 44 stand or fall together.

Arguments

I. Whether Claims 1, 21, 27, 32, 39, 40 and 46 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes.

Before considering this issue in detail, the Bhamidipati system is summarized for discussion purposes. Bhamidipati discloses a pipeline system having multiple stages. A decoupling queue is provided to decouple a pipeline stage N from a next stage N+1.[25] The decoupling queue operates so that during a given clock cycle, pipeline stage N may store an instruction to a location within the decoupling queue, and pipeline stage N+1 may retrieve an instruction from the same, or a different, location within this queue.[26] A write pointer is used to point to the storage location that will receive the next stored instruction. Similarly, a read pointer points to the storage location from which the next instruction will be retrieved.[27]

The Bhamidipati decoupling queue can operate in one of two modes. In a Write-Read (WR) mode, stage N stores an instruction into the decoupling queue before stage N+1 reads an instruction.[28] In contrast, when operating in Read-Write (RW) mode, stage N+1 first retrieves an instruction from the decoupling queue before stage N stores another instruction to the queue. The timing for these operations is illustrated by Bhamidipati Figure 9. Specifically, in WR mode, writing of an instruction occurs during period 902 followed by instruction

[25] Bhamidipati Abstract lines 5-8.
[26] Bhamidipati column 3 lines 20-23.
[27] Bhamidipati column 4 lines 12-14.
[28] Bhamidipati column 4 lines 7-10 and 18-24.

retrieval during period 904. In RW mode, the opposite occurs, with instruction retrieval being performed during duration 902 followed by the storing of an instruction during period 904.[29]

With the foregoing summary of the Bhamidipati system available for discussion purposes, the three sub-issues set forth above with respect to Issue I are considered in turn.

A.) Whether the Bhamidipati elements cited by the Examiner render Claims 1, 21, 27, 32, 39, 40 and 46 unpatentable in view of Hayes.

The Examiner's assertions concerning Bhamidipati are best considered in reference to Bhamidipati Figure 3. That Figure illustrates a circuit 502 that includes several pipeline stages 300, 302, 304, and 308. The Examiner cites these stages as disclosing Applicants' fetch stages.[30] The Examiner further asserts that Applicants' execution stages are disclosed by the execution element 106 of Figure 1.[31] Elements 300, 302, and 304 correspond to pipe stage F 100, of Figure 1, and instruction decode unit 308 of Figure 3 correlates to decode stage 102 of Figure 1.[32] Stage 104 may also be considered a fetch stage, since it precedes stage 106 cited by the Examiner as teaching Applicants' execution stages.

Next, the supposed teachings of Bhamidipati are considered in reference to the language of Applicants' representative Claim 1. According to this Claim,

[29] Bhamidipati column 5 lines 25-50.
[30] Paper Number 12, page 3, section 5, lines 11-14.
[31] Paper Number 12, page 3, section 5, lines 7-10

instructions are capable of advancing to a next stage within the pipeline fetch circuit independently of the times at which instructions advance to a next stage of execution within the execution circuit. This type of independent operation is not disclosed by the Bhamidipati fetch and execution stages, as cited by the Examiner. Specifically, the Bhamidipati stages 308 and 104 appear to operate in <u>lock-step</u> with execution stage 106. That is, once an instruction has been retrieved from decoupling queue 306 for processing within stage 308 of Figure 3, it appears that the way in which the instruction will advance will be entirely determined by the way instructions advance within the subsequent stages. There is no indication whatsoever in Bhamidipati that an instruction will advance from stage 308 to stage 104 while an instruction is stalled in stage 106.

To reiterate, according to the Examiner's analysis, the Bhamidipati decoupling queue 306 decouples two fetch stages 304 and 308. This use of a decoupling queue *between* two fetch stages *<u>does not</u>* provide a circuit that allows instructions to advance within the fetch stages independently of when instructions advance within the execution stage, as is described in Applicants' Claims. As previously discussed, this is because once an instruction enters fetch stage 308, any further advancement of this instruction appears to be directly controlled by whether instructions are advancing within the execution stage 106, and no indication of Bhamidipati is provided to the contrary.

The Examiner's interpretation of Bhamidipati as discussed above does not teach or suggest Applicants' system and method for yet another reason. In

[32] Bhamidipati column 3 lines 44-48.

Applicants' Claim 1, each instruction retained within the fetch circuit at a given time is in a respectively different stage of processing.[33] As defined in Applicants' Specification, a stage is a functional operation that is performed by a respective logic section of the IP.[34] Examples of such functional operations include the decoding of an instruction, generating an operand address, fetching an operand, performing manipulation on the operands, and etc.[35] This use of the term "stage" is consistent with the way in which the term is used in the art in general, and in Bhamidipati. For example, Bhamidipati describes each pipe stage as "...complet[ing] a part of an instruction".[36]

As discussed above, the Examiner states that Applicants' pipeline fetch circuit is taught by Bhamidipati stages 300, 302, 304, and 308.[37] Since the decoupling queue 306 interconnects stages 304 and 308 and affects the way these stages operate, the decoupling queue must be part of the Bhamidipati circuit 502 that supposedly teaches Applicants' fetch circuit. However, as discussed above, the decoupling queue is not a "stage" as that term is defined in Applicants' Specification and used in the art. Therefore, some of the instructions in the Bhamidipati circuit 502 are not in any "stage" at all, but are rather just stored within the decoupling queue. Thus, Bhamidipati does not teach Applicants' fetch circuit wherein _each_ of the instructions retained by the fetch circuit are in a _respectively different stage_ of processing. For this

[33] Claim 1 lines 10-13.
[34] Applicants' Specification page 12 lines 8-12.
[35] Specification page 12 lines 11-22.
[36] Bhamidipati column 1 lines 16-17.
[37] Paper 12 page 3, last full paragraph.

additional reason, Bhamidipati does not teach or suggest Applicants' claimed invention.

Next, the teaching of Hayes is considered. Hayes is a reference that pre-dates the Bhamidipati patent by more than two decades.[38] The Hayes reference discusses relatively primitive instruction pipelines that include a fetch stage coupled to an execution stage.[39] Hayes provides no teaching or suggestion whatsoever on allowing instructions to advance within fetch stages independently of how they advance within the execution stages. Hayes most certainly does not teach the aspects of Applicants' representative Claim 1 wherein the fetch and execution circuits are directly coupled <u>*yet independently operable*</u> with respect to the advancing of instructions.

Turning to the Examiner's analysis of Hayes, the Examiner cites the Hayes reference as teaching the concept that fetch and execution stages may be directly coupled.[40] This assertion is not understood for several reasons. First, according to the Examiner's analysis of Bhamidipati that is discussed above, the Bhamidipati fetch stages <u>are</u> directly interconnected to the Bhamidipati execution stages. Thus, the need to cite Hayes is not understood.

In addition to the foregoing, Hayes discusses a 1970s pipeline architecture that is an early precursor to the system described in Bhamidipati. In fact, the Bhamidipati background section begins with a discussion of a Hayes-type pipeline system wherein "[t]he pipe stages are connected one to the next to

[38] Copyright 1978 by McGraw-Hill, Inc.
[39] Hayes page 223, last full paragraph.
[40] Paper 12 page 5 line 2-3.

form a pipe, where instructions enter at one end, are processed through the stages, and exit at the other end."[41] The Bhamidipati description continues with the evolution of pipeline architectures, stating that "...in order to ensure the optimal performance of a pipeline, one method involves inserting queues in the pipeline to decouple these pipe stages."[42] This method is said to allow stages to carry on tasks independently. However, Bhamidipati also notes that this method imposes "...delay to perform either a read or write operation..." which can be "prohibitively lengthy in view of a diminishing clock cycle."[43] For this reason, Bhamidipati describes an adaptation of this decoupling queue mechanism that seeks to shorten this delay by using a queue supporting both read and write operations during a single clock cycle.[44]

To summarize, the Bhamidipati background section commences with a description of the Hayes 1970s pipeline architecture, and traces adaptations that culminate in the addition of the Bhamidipati decoupling queue. Therefore, it is Bhamidipati that is building upon the Hayes teachings to arrive at the decoupling queue mechanism rather than Hayes providing any teaching that could add to Bhamidipati. One skilled in the art would most certainly not be motivated to modify Bhamidipati with teachings of Hayes, and the cited combination of references is therefore improper, as will be discussed further below.

Even assuming one skilled in the art was somehow motivated to add some teaching of Hayes to the Bhamidipati system, it is not understood how this

[41] Bhamidipati column 1 line 17-19.
[42] Bhamidipati column 1 lines 24-26.
[43] Bhamidipati column 1 lines 41-43.

supposed teaching would motivate one skilled in the art to arrive at Applicants' claimed invention. Hayes provides no teaching whatsoever on how to obtain a pipeline wherein the fetch and execution circuits are directly coupled _yet at the same time independently operable_ in the manner claimed by the Applicants. Neither Bhamidipati or Hayes, alone or in combination, teaches or suggests this type of system, and the Bhamidipati elements cited by the Examiner do not render Claims 1, 21, 27, 32, 40 and 46 unpatentable in view of Hayes.

B.) Whether any other elements of Bhamidipati render Claims 1, 21, 27, 32, 39, 40 and 46 unpatentable in view of Hayes.

The foregoing analysis focuses on the Bhamidipati elements and embodiments that are cited by the Examiner in Paper 12 as teaching Applicants' claimed invention. However, Bhamidipati discusses alternative embodiments wherein decoupling queues may be inserted between other pipeline stages.[45] These additional embodiments are considered for completeness.

According to one alternative embodiment, a decoupling queue may be inserted between the Bhamidipati fetch and execution stages rather than between two of the fetch stages, as was considered above. This results in a situation wherein the last fetch stage 104 provides an instruction to be stored within the decoupling queue 306. Sometime later, the first execution stage retrieves this instruction for execution. This embodiment does not teach or suggest Applicants' system for providing instructions _directly_ from the fetch

[44] Bhamidipati column 1 lines 56-61.

stages to the execution stages, as claimed by representative Claim 1 and the remaining Claims.[46]

The advantages provided by Applicants' claimed invention as compared to that of the Bhamidipati system become apparent when considering Bhamidipati Figure 9. Assume that the Bhamidipati decoupling queue 306 is empty such that the system is operating in read-write mode[47]. In this mode, an instruction is stored to the decoupling queue 306 during time period 902. The instruction is subsequently retrieved from the decoupling queue by the first execution stage during time period 904[48]. This instruction is therefore not available for processing by the first execution stage until time 908. Thus, in the situation wherein the queue is empty and the execution stage is waiting to receive another instruction from the last fetch stage, virtually an _entire clock cycle is wasted_ because the decoupling queue is inserted between the fetch and execution stages. As previously discussed, this problem of using decoupling queues is recognized by Bhamidipati, itself:

> "As a processor further splits up its pipe stages and increases its clock speed, the duration of the decoupling queue's setup time and its delay to perform either a read or a write operation can become prohibitively lengthy in view of a diminishing clock cycle. When such overhead equals to [sic] the processor clock cycle, no further pipelining is useful in enhancing a processor's performance."[49]

---

[45] Bhamidipati column 3 lines 54 - 64.
[46] Claim 1 line 10.
[47] Bhamidipati column 4 lines 25-31.
[48] Bhamidipati column 4 lines 25-38.
[49] Bhamidipati column 1 lines 39-45.

Although Bhamidipati seeks to minimize this discussed problem by providing a de-coupling queue that is capable of both storing and retrieving an instruction within the same clock cycle, the problem is not entirely eliminated. In those situations wherein the decoupling queue has become empty, a delay of more than half a clock cycle occurs between the time an instruction is available from the last fetch stage until the time it is received by the first execution stage. This delay is not insignificant in high-speed pipeline architectures.

Applicants' system of representative Claim 1 addresses this type of problem by providing instructions directly from the fetch circuit to the execution circuit without utilization of a storage device such as queue.[50] This direct coupling is provided by the 3Y1 register 250 of Applicants' Figure 11, which implements the last fetch stage, and which is directly interconnected to provide an instruction to the first execution stage embodied by instruction decode dispatch logic 124 of Figure 7. In this manner, an instruction may be provided without delay from Applicants' last fetch stage to the first execution stage. Instead of employing a storage device to decouple these two stages, the control circuit shown in Applicants' Figures 10 and 11 is employed to provide the decoupling function.[51]

The alternative Bhamidipati embodiment discussed in the above paragraphs provides a system wherein the fetch and execution stages are not directly coupled. Instead the fetch and execution stages are indirectly coupled through a decoupling queue.

It may be appreciated from the foregoing that the alternative embodiment of Bhamidipati does not alone teach Applicants' invention for directly coupling independently operable fetch and execution stages of a pipeline. Next, it may be considered whether the Bhamidipati alternative embodiment provides this teaching when coupled with Hayes.

As previously discussed, Hayes provides an early pipeline architecture from the 1970s. Bhamidipati expressly discusses how and why this type of early Hayes pipeline architecture should be modified to obtain the system of Bhamidipati. One skilled in the art would not be motivated by Hayes to modify Bhamidipati when Bhamidipati is expressly teaching how to modify Hayes. Thus, the alternative Bhamidipati embodiment discussed above does not render the Claims unpatentable with, or without, the addition of Hayes.

Finally, yet another characterization of the Bhamidipati system must be considered for completeness sake. In Paper 15, the Examiner states that the decoupling queue of Bhamidipati might be characterized as _part of_ the Bhamidipati execution or fetch circuit rather than as a separate element that links these types of stages.[52] For example, a decoupling queue could be inserted before the execution stages 106 of the pipeline circuit of Figure 1. The decoupling queue and execution stages 106 may then be considered an "execution circuit" that is directly coupled to a fetch circuit, and which is then said to teach Applicants' claimed invention.

[50] Claim 1 line 10.
[51] See Specification pages 21-25 for a discussion of these Figures.
[52] Paper Number 15.

The foregoing characterization does not teach or suggest Applicants' claimed invention for reasons that are similar to those described in Section I above. Assume, for example, that the decoupling queue is included within the Bhamidipati execution circuit as described in the previous paragraph. This decoupling queue stores one or more instructions that are awaiting retrieval by the first execution stage. This queue does not perform any type of functional operations on the instructions. Therefore, this decoupling queue cannot be categorized as a "stage" of execution as that term is used in Applicants' Specification and in the relevant arts because the decoupling queue is not performing any functional operations on the instruction.

Because the Bhamidipati decoupling queue is not a "stage of execution" as that term is used either in Applicants' Specification or in Bhamidipati, the Examiner's alternative characterization of Bhamidipati does not teach or suggest Applicants' claimed invention. Applicants' representative Claim 1 describes an execution circuit wherein each instruction in the execution circuit is in a *respectively different stage* of execution. This is not true in the Bhamidipati "execution circuit" of the current analysis, wherein at least one instruction within this circuit will not be in a stage of execution at all, but instead will be stored within the decoupling queue. Viewed a different way, if the decoupling queue is considered part of the first execution stage, two instructions within the execution circuit will be in this same first execution stage.

To summarize, if the Bhamidipati decoupling queue is considered part of the execution circuit, all instructions within this execution circuit will not be within a respectively different stage of execution as claimed by Applicants' Claims, and Bhamidipati does not teach or suggest Applicants' claimed invention. As discussed above in Section I, a similar analysis applies if the decoupling queue is instead considered part of a Bhamidipati "fetch circuit".

For all of the foregoing reasons, none of the embodiments, or alternative characterizations of Bhamidipati, render Applicants' Claims 1, 21, 27, 32, 39, 40, and 46 unpatentable in view of Hayes.

<u>C.) Whether the Examiner has set forth a prima facie case for obviousness with respect to Claims 1, 21, 27, 32, 39, 40 and 46.</u>

As discussed previously, the 1970-version pipeline illustrated in Hayes is a precursor to that described in Bhamidipati over two decades later. In fact, Bhamidipati discusses the Hayes-type pipeline, including its deficiencies, in the background of the Bhamidipati invention.[53] Bhamidipati then offers an improvement to the Hayes pipeline that allows pipeline stages to be decoupled to ensure more optimal performance.[54] The Examiner never-the-less cites Hayes for teaching that a fetch stage may be directly coupled to an execution stage.[55] It is submitted that one skilled in the art would not be motivated to modify Bhamidipati with any teaching from Hayes, particularly when Bhamidipati is expressly providing an *<u>improvement to the Hayes pipeline</u>*.

[53] Bhamidipati column 1 lines 12-25 discuss.
[54] Bhamidipati column 1 lines 24-35.
[55] Paper 12 page 6 last paragraph.

The Examiner states that one skilled in the art would be motivated to make the cited combination because it would result in implementation of a standard, well-known, and easy to implement pipeline.[56] As previously stated, at the time of Bhamidipati, the Hayes system was over two decades old. Bhamidipati et al. was well aware of Hayes-type systems, as evidenced by the Bhamidipati discussion of the prior art. If one skilled in the art would have been motivated to incorporate additional aspects of Hayes into the Bhamidipati system, presumably Bhamidipati et al. would have done so. However, this was not the case. Because there is absolutely no suggestion or motivation, either in the references themselves, or in the knowledge available to one skilled in the art, to modify Bhamidipati with Hayes, the Examiner has failed to set forth a prima facie case of obvious.[57]

Further to the foregoing point, in expressly discussing a Hayes-type system, then providing a solution to the Hayes deficiencies, Bhamidipati is <u>*teaching away*</u> from the direct coupling of stages used in the Hayes system and Applicants' claimed invention. Teaching away has long been considered an indication that the cited combination of references is improper.[58]

Not only does Bhamidipati teach away from the cited combination of references and from Applicants' claimed invention, but Hayes does so as well. As discussed above, Hayes teaches a system wherein instructions advance in lock step between all stages of the pipeline, including between the fetch and

[56] Paper 12 page 5, lines 5-8.
[57] *In re Oetiker* 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992).
[58] *In re Gurley*, 27 F.3d 551, 31 USPQ 2d 1130, 1131, (Fed. Cir. 1994).

execution stages. While Hayes briefly notes the problems associated with this lock-step mechanism that are caused by varying fetch and execution times, the only solution Hayes offers involves allowing some instructions to by-pass some of the pipeline stages.[59] Thus, although Hayes recognizes some of the problems solved by Applicants' invention, Hayes actually *teaches away* from Applicants' claimed invention by suggesting that instructions should *by-pass* pipeline stages rather than being passed *directly between* those stages.

To summarize, there is absolutely no motivation to add any elements of Hayes to Bhamidipati. The Examiner has therefore failed to set forth a prima facie case of obviousness with respect to Claims 1, 21, 27, 32, 39, 40, and 46, and for this reason alone, these Claims should be allowed.[60] Moreover, even if elements of Hayes were added to Bhamidipati, it would not teach or suggest a system that allows fetch and execution stages of a pipeline to be both directly coupled and independently operable. For this additional reason, these Claims are patentable over the cited combination of references.

II. Whether Claims 1, 3-4, 21, 27, 32, 39, 40 and 46 are anticipated under 35 USC § 102(e) by Bhamidipati.

As an alternative rejection to that discussed in Section I, above, the Examiner further rejects the Claims as anticipated under 35 USC §102(e) based on Bhamidipati. This rejection under 35 USC §102(e) is set forth in Paper 16 dated 10/18/2004 which re-opens prosecution of the Claims after Appeal, and

[59] Hayes page 24, last paragraph.
[60] *In re Oetiker*, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992).

which further maintains the original rejection of these Claims under 35 USC §103(a), as discussed in Section I above.[61]

In regards to this alternative rejection, the Examiner states that Bhamidipati elements 104 and 106 teach Applicants' execution circuit, and Bhamidipati element 102 teaches Applicants' fetch circuit. The Examiner further asserts that the decoupling queue 306, which is implemented "as the end of the last decode stage" (i.e., the last fetch stage) teaches the capability of allowing an instruction to advance to a next stage of execution within the fetch circuit independently of the times instructions are advancing within the execution circuit.[62]

The Examiner's analysis set forth above can be interpreted in two ways, both of which were analyzed for completeness in Section I above. According to a first interpretation, the queue, which is said to be implemented "as the end of the last decode stage"[63], is an integral part of that fetch stage. As such, when an instruction is being executed within that last Bhamidipati fetch stage, and further when another instruction resides within the queue that is integral to this stage, the last fetch stage retains multiple instructions. Therefore, according to this interpretation, Bhamidipati does not teach Applicants' claimed pipeline fetch circuit wherein each of the predetermined instructions that may be retained by the fetch circuit is in a respectively different stage of processing.

[61] Paper 16 page 2, Section 2, and page 26 Section 7.
[62] Paper 16, page 3 and first paragraph of page 4.
[63] Paper 16, page 4 line 5.

According to a second interpretation, the queue is actually a separate stage of the fetch circuit. This appears to be the interpretation that is being set forth by the Examiner's assertion that "element 306 [is]... shown as [its] own stage of processing"[64]. However, this interpretation does not coincide with the definition of "stage" as set forth in Applicants' Specification, nor is it consistent with the description set forth in Bhamidipati itself. As previously discussed, Applicants' Specification describes a stage as a <u>functional operation</u> that is performed by a respective logic section of the IP when executing an instruction. Examples of such functional operations include the decoding of an instruction, generating an operand address, fetching an operand, performing manipulation on the operands, and etc.[65] The retaining of an instruction in a queue is not this type of a functional operation performed when executing an instruction.

Bhamidipati describes a stage in a manner that is similar to the way that this term is used in Applicants' Specification. In particular, Bhamidipati states:

> "Each step in the pipeline, or a pipe stage, *<u>completes a part of an instruction.</u>*"[66]

This passage makes clear that a stage involves execution of a part of the instruction. The mere storing of an instruction in a queue <u>does not complete</u> part of the instruction, and thus the <u>queue is not a stage</u> as the term is defined in Bhamidipati. This interpretation is confirmed by further description as follows:

[64] Paper 16, page 3 lines 19-20.
[65] For example, see Applicants' Specification page 12 lines 8-22.
[66] Bhamidipati, column 1 lines 16-17.

> "...a decoupling queue is provided *to decouple* at least one of the mentioned pipe *stages* from another..."[67]

This passage appears to make clear that Bhamidipati does not describe the decoupling queue as being a stage itself, but instead as an entity inserted between two stages. A similar description discusses the Bhamidipati pipeline of Figure 3 as follows:

> "Blocks 300, 302 and 304 can together be considered as pipe stage F 100 as shown in FIG. 1...instruction decode unit 308, or pipe stage D 102 ...does not need to stall for completion of pipe stage F 100. In other words, operations of *pipe stage F 100 and pipe stage D 102 have been decoupled* by use of decoupling *queue* 306. "[68]

This description states that blocks 300-304 comprise pipe stage 100 of FIG. 1, block 308 comprises pipe stage 102, and decoupling queue is used to decouple pipe stage 100 from stage 102. Thus, it appears that all of the blocks of Bhamidipati FIG. 3 except the decoupling queue 306 are considered part of a stage of the pipeline. This interpretation is again reiterated in the Bhamidipati Claims, which describe:

> "a plurality of pipe stages; and...a decoupling queue to decouple at least one of said pipe stages from another"[69]

[67] Bhamidipati column 1 lines 56-58, emphasis added.
[68] Bhamidipati column 3 lines 45-53, emphasis added.
[69] Bhamidipati Claim 1 lines 5-7, Claim 10 lines 4-6, and Claim 19 lines 5-8, emphasis added.

The Bhamidipati Claims thus make clear that the decoupling queue is not considered a stage of the Bhamidipati pipe, but rather logic that decouples one stage from another.

To summarize, both Bhamidipati and Applicants' Specification use the term "stage" to describe some type of processing step or operation that is performed on the instruction. This is consistent with how that term is used in the art. For example, Hayes describes a pipe stage as follows:

> "A pipeline processor consists of a sequence of *processing circuits*, called *segments or stages*, through which a stream of operands can be passed;...*Partial processing of the operands* takes place in *each segment*,... [70]

Thus, Hayes also describes a pipeline stage as performing processing operations. Merely storing of an instruction in a queue is not performing any processing step, and the Bhamidipati decoupling queue is therefore not considered a stage.

Because the decoupling queue is not a pipeline stage at all, but rather a queue that decouples the fetch circuits from the execution circuits, Bhamidipati does not teach Applicants' system of Claim 1 wherein instructions are provided directly from a fetch to an execution circuit, and each instruction is in a respectively different stage of execution.

[70] Hayes page 592, Section 7.2.1, first three lines, emphasis added.

In conclusion, regardless of which of the Examiner's interpretations is considered in regards to the alternative §102 rejection that is raised in Paper 16, Bhamidipati does not teach a system containing all of the following capabilities:

- a fetch circuit that is coupled to provide instructions <u>directly</u> to an execution circuit;
- a circuit wherein each of the instructions in both the fetch and execution circuits is in a <u>different stage</u> of execution; and
- a circuit wherein the instructions advance within the fetch circuit <u>independently</u> of when the instructions advance within the execution circuit.

Because Bhamidipati does not teach each and every element of Applicants' system, this rejection is improper and should be withdrawn.

<u>III. Whether Claims 2-6, 22-25, 28-30, 33-38, 41-43, and 45 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes, and further in view of Kyker.</u>

Each of Applicants' Claims 2-6, 22-25, 28-30, 33-38, 41-43, and 45 depends from an independent Claim considered in the foregoing Section I, and is patentable over the cited prior art for the reasons discussed above in Section I. These Claims describe various other aspects of Applicants' invention that the Examiner asserts are taught by Kyker.

Kyker describes an improved branch recovery mechanism that is used to load correct micro operations into a pipeline after a branch misprediction has occurred.[71] Nothing in Kyker adds anything to Bhamidipati or Hayes to teach Applicants' invention that allows fetch and execution stages to be both directly coupled and independently operable. Applicants' Claims 2-6, 22-25, 28-30, 33-38, 41-43, and 45 are therefore patentable over Bhamidipati in view of Hayes, and further in view of Kyker.

IV. Whether Claims 2, 5-6, 22-25, 28-30, 33-38, 41-43, and 45 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Kyker.

In Paper 16, the Examiner raises this alternative rejection in regards to Claims 2, 5-6, 22-25, 28-30, 33-38, 41-43, and 45 based on Bhamidipati and Kyker.[72] Each of these Claims depends from an independent Claim that is allowable over Bhamidipati for the reasons discussed in the foregoing Section II in regards to the alternative rejection. Nothing in Kyker adds anything to Bhamidipati to teach Applicants' invention that allows fetch and execution stages to be both directly coupled and independently operable, and these Claims are therefore patentable over Bhamidipati in view of Kyker.

V. Whether Claim 7 is unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes and Kyker, and further in view of Alferness.

[71] Kyker column 3 lines 48-63.
[72] Paper 16 page 11.

Dependent Claim 7 provides a pipeline execution circuit that includes:

> "a microcode-controlled sequencer to control execution of extended stages of execution of extended-mode ones of the instructions, wherein during said extended stages of execution, ones of the instructions being executed by said pipeline execution circuit are not advancing to a next stage of execution within said pipeline execution circuit, and wherein said first selection circuit includes a control circuit to allow an instruction to enter said pre-decode stage of processing while said extended-mode ones of the instructions are not advancing to a next stage of execution within said pipeline execution circuit.

According to the embodiment of Claim 7, Applicants' pipeline includes a microcode-controlled sequencer. This sequencer, which is shown in block 142 of Figure 7, executes microcode that controls the various logic sections of the processor during the extra extended-mode stages of execution for special extended-mode instructions. These extended-mode instructions are instructions that do not complete execution during the standard stages of pipeline execution. The timing associated with these extended-mode stages is shown in Figure 6[73].

Turning next to the rejection of Claim 7, this Claim depends from Claim 5, which is considered in Section III, above. Claim 7 is patentable over the cited prior art for the reasons discussed above in Section III in regards to Claim 5. Moreover, in regards to the aspects of Claim 7 involving extended-mode instructions, it may be noted that the Bhamidipati system does not include extended-mode instructions. Instead, in Bhamidipati, it appears that an execution stage X of any given instruction requires only a single clock cycle to

[73] Waveform 104 of Figure 6 illustrates extended stages 112.

complete.[74] Thus, one skilled in the art would not be motivated by anything in the references themselves or the prior art as a whole to add any type of microsequencer for controlling extended-mode instructions to the design of Bhamidipati. Moreover, by citing four separate unrelated references, the Examiner is attempting to piece together the claimed invention in hindsight, something that has long been held impermissible.[75] The Examiner has failed to set forth a prima facie case of obviousness with respect to Claim 7, and for this additional reason, Claim 7 is patentable over the cited combination of references[76].

<u>VI. Whether Claim 7 is unpatentable under 35 USC §103(a) over Bhamidipati in view of Kyker, and further in view of Alferness.</u>

In Paper 16, the Examiner raises an alternative rejection in regards to Claim 7[77] based on Bhamidipati, Kyker, and Alferness. Claim 7 depends from Claim 5, and is allowable for the reasons discussed above in Section IV in regards to Clam 5. As previously discussed, Claim 7 further describes the aspect of Applicants' invention wherein the execution circuit includes a microcode-controlled sequencer to control execution of extended-mode instructions. Because the instruction set of Bhamidipati does not include these types of instructions, one skilled in the art would not be motivated by anything in

[74] Bhamidipati column 2 lines 50-52, which discusses that each instruction takes four clock cycles to complete the four stages F, D, A, and X of the pipeline, and each clock cycle, the stages are operating on four different instructions.
[75] *In re Gorman*, 933 F.2d 982, 986, 18 USPQ2d 1885, 1888 (Fed.Cir.1991).
[76] *In re Oetiker*, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992).

the cited references themselves or the prior art as a whole to add any type of microsequencer for controlling extended-cycle instructions to the design of Bhamidipati. The Examiner has failed to set forth a prima facie case of obviousness with respect to Claim 7 in this alternative rejection, and for this additional reason, Claim 7 is patentable over the cited combination of references[78].

VII. Whether Claims 26, 31, and 44 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Hayes and further in view of Alferness.

Representative Dependent Claim 26 provides for a pipeline circuit

> "...wherein at least one of the execution stages includes a microcode-controlled sequencer to control execution of extended-mode instructions, and wherein during some stages of execution of the extended-mode instructions, instructions are not advancing within the execution circuit."

Claims 31 and 44 include aspects similar to Claim 26. These Claims relate to Applicants' microcode sequencer shown in block 142 of Figure 7 and discussed above. This sequencer executes microcode that controls the various logic sections of the processor during the extra "extended-mode" stages of special extended-mode instructions. When an extended-mode instruction enters the execution stages of the pipeline, the instruction does not advance to the 2X stage of the pipeline until multiple clock cycles have elapsed, as shown by waveform 104 of Figure 6.

[77] Paper 16 page 22.
[78] *In re Oetiker*, 977 F.2d 1443, 1445, 24 USPQ2d 1443, 1445 (Fed. Cir. 1992).

Applicants' Claims 26, 31, and 44 depend from various Claims discussed above in Sections I and III, and are patentable over this rejection for the reasons set forth above in these Sections. These Claims further describe the aspect wherein Applicants' execution stages include a microcode-controlled sequencer to control execution of extended-mode instructions.

As discussed previously, the Bhamidipati instruction set does not include extended-mode instructions that require more than a clock cycle to advance between any two execution stages. Instead, all Bhamidipati instructions appear to be "single-cycle" instructions that require a single clock cycle to complete any given stage of execution.[79] Thus, one skilled in the art would not be motivated by anything in the references themselves or the prior art as a whole to add any type of microsequencer for controlling extended-mode instructions to the design of Bhamidipati. The Examiner has therefore failed to set forth a prima facie case of obviousness with respect to Claims 26, 31, and 44, and these Claims are patentable over the cited combination of references for this additional reason.

VIII. Whether Claims 26, 31, and 44 are unpatentable under 35 USC §103(a) over Bhamidipati in view of Alferness.

[79] Bhamidipati column 2 lines 50 through 52, which discusses that each instruction takes four clock cycles to complete the four stages F, D, A, and X of the pipeline. Although this discussion relates to Figure 1 designated as "prior art", the Bhamidipati system of Figure 3 is described as being incorporated into the pipeline of Figure 1 in column 3 lines 44-49.

In Paper 16, the Examiner raises an alternative rejection in regards to Claims 26, 31, and 44 based on Bhamidipati and Alferness.[80] Each of these Claims depends from an independent Claim that is allowable over Bhamidipati for at least the reasons discussed in the foregoing Section II. These Claims further describe the aspect wherein Applicants' execution stages include a microcode-controlled sequencer to control execution of extended-mode instructions.

As previously discussed, the instruction set of Bhamidipati does not appear to include extended-mode instructions. Thus, one skilled in the art would not be motivated by anything in the references themselves or the prior art as a whole to add a microsequencer for controlling extended-mode instructions to the Bhamidipati system. The combination of elements of Alferness with the Bhamidipati system is therefore improper, and the Examiner has failed to set forth a prima facie case of obviousness with respect to Claims 26, 31, and 44. These Claims are patentable over the cited combination of references for this additional reason.

[80] Paper 16 page 23.

Conclusion and Request for Relief

Applicants' Claims 1-7 and 21-46 are patentable over the cited combination of references. None of the references, alone or in combination, teach or suggest an instruction pipeline that allows instructions to advance within the fetch stages independently of the times at which they advance within the execution stages, while also allowing instructions to be provided directly by a fetch stage to an execution stage so that pipeline performance is optimized. Moreover, the Examiner has failed to set forth a prima facie case of obviousness because one skilled in the art would not be motivated to modify Bhamidipati with Hayes, particularly when Bhamidipati explicitly sets forth an improvement to the early Hayes-type design. Additionally, in regards to the alternative rejection, Bhamidipati alone does not anticipate Applicants' independent Claims because this reference does not teach each and every element of the Claims.

The Examiner has further failed to set forth a prima facie case of obviousness with respect to Claims 7, 26, 31 and 44 for the additional reason that one skilled in the art would not be motivated to add a microsequencer of the type described in Applicants' Claims to Bhamidipati, which includes an instruction set having single-cycle instructions. For all of these reasons, Claims 1-7 and 21-46 are patentable over the cited references. It is therefore respectfully requested that the rejection of the Claims be overturned, and the Claims be passed to issue.

Respectfully submitted,

Beth L McMahon 2/10/05

Beth L. McMahon
Attorney for Applicants
Reg. No. 41,987
Telephone No. 651-635-7893
UNISYS Corporation
M.S. 4773
PO Box 64942
St. Paul, MN 55164-0942

Appendix A

Presented is a clean set of Claims 1-7 and 21-46 as last amended April 30, 2003.

Claim 1 (Previously Amended):

1. For use in an instruction processor that executes instructions included in a predetermined instruction set at an execution rate determined by a system clock signal, a synchronous instruction pipeline, comprising:

a pipeline execution circuit to process a first predetermined number of instructions simultaneously, each of said first predetermined number of instructions being in a respectively different stage of execution within said pipeline execution circuit, instructions being capable of advancing to a next stage of execution within said pipeline execution circuit at a time determined by the system clock signal; and

a pipeline fetch circuit coupled to provide each of the first predetermined number of instructions directly to said pipeline execution circuit, the pipeline fetch circuit to retain a second predetermined number of instructions simultaneously, each of said second predetermined number of instructions being in a respectively different stage of processing within said pipeline fetch circuit, an instruction being capable of advancing to a next stage of execution within said pipeline fetch circuit at a time determined by the system clock signal and independently of the times at which instructions advance to a next stage of execution within said pipeline execution circuit.

Claim 2 (Original):

2. The synchronous instruction pipeline of Claim 1, wherein said pipeline fetch circuit includes an instruction queue to store a predetermined maximum number of the instructions that are each ready to be processed by said pipeline fetch circuit.

Claim 3 (Original):

3. The synchronous instruction pipeline of Claim 1, wherein said pipeline fetch circuit includes a pre-decode logic circuit to generate pre-decode signals for an instruction that is in a pre-decode stage of processing within said pipeline fetch circuit, and wherein an instruction can enter said pre-decode stage of processing independently of the movement of instructions through said pipeline execution circuit.

Claim 4 (Original):

4. The synchronous instruction pipeline of Claim 3, wherein said pipeline fetch circuit includes a decode logic circuit coupled to said pre-decode logic circuit to generate decode signals for an instruction that is in a decode stage of processing within said pipeline fetch circuit, and wherein an instruction can enter said decode stage of processing from said pre-decode stage of processing independently of the movement of instructions through said pipeline execution circuit.

Claim 5 (Original):

5. The synchronous instruction pipeline of Claim 4, wherein said pipeline fetch circuit includes a first selection circuit coupled to said pre-decode logic circuit to allow an instruction to be received by said pre-decode logic circuit at a time determined by the system clock signal if said decode logic circuit is available to accept an instruction currently being executed by said pre-decode logic circuit.

Claim 6 (Original):

6. The synchronous instruction pipeline of Claim 5, wherein said pipeline fetch circuit includes a second selection circuit coupled to said decode logic circuit to allow an instruction to enter said decode stage of execution at a time determined by the system clock signal if said decode logic circuit is not processing another instruction.

Claim 7 (Previously Amended):

7. The synchronous instruction pipeline of Claim 5, wherein said pipeline execution circuit includes a microcode-controlled sequencer to control execution of extended stages of execution of extended-mode ones of the instructions, wherein during said extended stages of execution, ones of the instructions being executed by said pipeline execution circuit are not advancing to a next stage of execution within said pipeline execution circuit, and wherein said first selection circuit includes a control circuit to allow an instruction to enter said pre-decode

stage of processing while said extended-mode ones of the instructions are not advancing to a next stage of execution within said pipeline execution circuit.

Claim 21 (Previously Added)

21. For use in an instruction processor, a synchronous pipeline circuit, comprising:

an execution circuit to provide a first predetermined number of execution stages, each being capable of performing a respective processing operation on a respective instruction; and

a fetch circuit coupled to the execution circuit to provide a second predetermined number of fetch stages, each fetch stage being capable of performing a respective pre-execution operation on a respective instruction, the fetch circuit to transfer each instruction processed by the fetch circuit directly from one of the fetch stages to one of the execution stages, ones of the instructions processed within the fetch stages being capable of advancing to different available fetch stages independently of whether instructions are advancing within the execution stages.

Claim 22 (Previously Added)

22. The pipeline circuit of Claim 21, wherein one of the fetch stages includes instruction address generate logic to predict which instructions are to enter the fetch stages.

Claim 23 (Previously Added)

23. The pipeline circuit of Claim 22, wherein the instruction address generate logic includes a circuit to clear ones of the fetch stages in response to a determination that instruction execution was re-directed.

Claim 24 (Previously Added)

24. The pipeline circuit of Claim 21, and further including

a memory to store instructions;

a queue coupled to the memory to temporarily store at least one instruction fetched from the memory; and

a circuit coupled to the queue and to at least one of the fetch stages to fetch an instruction from the queue for presentation to at least one of the fetch stages.

Claim 25 (Previously Added)

25. The pipeline circuit of Claim 24, wherein the circuit coupled to the queue is capable of retrieving instructions from the queue for presentation to the at least one of the fetch stages regardless of whether instructions are advancing within the execution stages.

Claim 26 (Previously Added)

26. The pipeline circuit of Claim 21, wherein at least one of the execution stages includes a microcode-controlled sequencer to control execution of

extended-mode instructions, and wherein during some stages of execution of the extended-mode instructions, instructions are not advancing within the execution circuit.

<u>Claim 27 (Previously Added)</u>

27. A synchronous instruction pipeline circuit for processing instructions within a data processing system, comprising:

a first predetermined number of fetch stages to simultaneously process at least a first predetermined number of instructions;

a second predetermined number of execution stages to simultaneous process a second predetermined number of instructions, each received directly from one of the fetch stages; and

wherein at least one of the fetch stages is capable of providing an instruction to a different one of the fetch stages that is ready to receive an instruction irrespective of movement of instructions between the execution stages.

<u>Claim 28 (Previously Added)</u>

28. The pipeline circuit of Claim 27, wherein one of the fetch stages includes address generate logic to predict which instructions are to enter the fetch stages for processing.

Claim 29 (Previously Added)

29. The pipeline circuit of Claim 28, wherein the address generate logic includes a circuit to flush one or more instructions from the fetch stages if it is determined that a misprediction occurred.

Claim 30 (Previously Added)

30. The pipeline circuit of Claim 27, and further including:

a memory; and

a storage device coupled to one of the fetch stages and to the memory to store instructions retrieved from the memory, wherein a predetermined number of instructions may be stored within the storage device regardless of whether instructions are advancing within the fetch stages.

Claim 31 (Previously Added)

31. The pipeline circuit of Claim 27, wherein one of the execution stages includes a microcode sequencer to execute predetermined ones of the instructions in a manner that may temporarily affect movement of instructions within the execution stages.

Claim 32 (Previously Added)

32. A synchronous instruction pipeline to execute instructions, comprising:

an execution circuit having a first predetermined number of execution stages to execute a first predetermined number of instructions substantially simultaneously; and

a fetch circuit having a second predetermined number of fetch stages to perform pre-execution operations on at least a second predetermined number of instructions substantially simultaneously, one of the fetch stages being coupled to provide each instruction processed by the fetch circuit directly to one of the execution stages, at least one of the at least second predetermined number of instructions being capable of advancing between different ones of the fetch stages regardless of whether an instruction is being transferred by the fetch circuit to the execution circuit.

Claim 33 (Previously Added)

33. The pipeline of Claim 32, wherein the fetch circuit includes an instruction address generate section to determine which instructions are to enter the fetch circuit.

Claim 34 (Previously Added)

34. The pipeline of Claim 33, wherein the instruction address generate section includes a circuit to remove instructions from the fetch circuit during a pipeline flush operation.

Claim 35 (Previously Added)

35. The pipeline of Claim 32, and further including:

a memory to store instructions; and

a queue coupled to store instructions from the memory, the queue further being coupled to provide an instruction to the fetch circuit if one of the fetch stages is available and irrespective of whether an instruction is being provided from the fetch circuit to the execution circuit.

Claim 36 (Previously Added)

36. The pipeline of Claim 35, and further including a circuit coupled to the queue to allow an instruction to be stored to the queue independently of whether an instruction is advancing within the fetch circuit.

Claim 37 (Previously Added)

37. The pipeline of Claim 36, wherein the circuit allows a predetermined maximum number of instructions to be stored to the queue independently of whether an instruction is advancing within the fetch circuit.

Claim 38 (Previously Added)

38. The pipeline of Claim 37, wherein the one of the fetch stages includes a circuit to allow retrieval of an instruction from either the memory or from the queue.

Claim 39 (Previously Added)

39. The pipeline of Claim 32, wherein the fetch circuit includes a circuit that allows instructions to advance within the second predetermined number of fetch stages if one of the execution stages is performing a predetermined function.

Claim 40 (Previously Added)

40. A method of processing instructions within a synchronous pipeline of an instruction processor, comprising:

a.) performing pre-execution operations on a first predetermined number of instructions substantially simultaneously within a first predetermined number of fetch stages of the pipeline;

b.) executing a second predetermined number of instructions substantially simultaneously within a second predetermined number of execution stages of the pipeline, wherein each of the second predetermined number of instructions were received directly from one of the fetch stages; and

c.) allowing one or more of the first predetermined number of instructions to advance between ones of the fetch stages independently of whether any of the second predetermined number of instructions are advancing between ones of the execution stages.

Claim 41 (Previously Added)

41. The method of Claim 40, and further including:

fetching an instruction from a memory;

storing the instruction within a queue; and

retrieving the instruction from the queue to undergo a pre-execution operation within a predetermined one of the fetch stages.

<u>Claim 42 (Previously Added)</u>

42. The method of Claim 41, wherein at least one of the storing and the retrieving step is performed independently of whether instructions are advancing between ones of the execution stages.

<u>Claim 43 (Previously Added)</u>

43. The method of Claim 42, wherein ones of the steps are repeated for multiple instructions.

<u>Claim 44 (Previously Added)</u>

44. The method of Claim 40, wherein one of the execution stages includes a microcode-controlled sequencer for executing extended-mode instructions, and further including executing one of the extended-mode instructions in a manner that temporarily delays the advancing of instructions between ones of the execution stages.

<u>Claim 45 (Previously Added)</u>

45. The method of Claim 40, and further including:

providing an indication that one or more predetermined operations are occurring within one or more of the execution stages; and

in response to the indication, allowing instructions to advance within the fetch stages.

Claim 46 (Previously Added)

46. A pipeline circuit for use in an instruction processor, comprising:

instruction fetch means for performing pre-execution operations on a first predetermined number of instructions substantially simultaneously within a first predetermined number of fetch stages;

instruction execution means for executing a second predetermined number of instructions substantially simultaneously within a second predetermined number of execution stages, each of the second predetermined number of instructions being received directly from one of the fetch stages; and wherein

the instruction fetch means includes means for allowing at least one of the first predetermined number of instructions to advance within the fetch stages irrespective of whether instructions are advancing within the execution stage.